PATENT ABSTRACTS OF JAPAN

(11)Publication number : 06-243112
(43)Date of publication of application : 02.09.1994

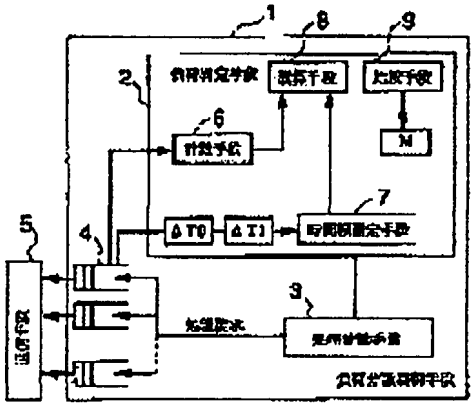
(51)Int.Cl. G06F 15/16
G06F 9/46
G06F 13/00

(21)Application number : 05-030613
(22)Date of filing : 19.02.1993

(71)Applicant : SEIKO EPSON CORP
(72)Inventor : NAGASAKA FUMIO

(54) MULTIPROCESSOR DEVICE

(57)Abstract:
PURPOSE: To attain the optimum dispersion of load by detecting the load on other processors in a multiprocessor parallel processing system working in a network environment where different device types are mixed.
CONSTITUTION: A queue 4 and a load deciding means 2 are provided for each processor when the processing units are dispersed to other processors and the processing time is shortened by the parallel execution by the processors. The means 2 performs an arithmetic coperation based on the length of the queue 4 and the results ΔT0 and ΔT1 of hitherto processing time and returns the expected value M of the end time of the next processing. A processing dispersion means 3 distributes the processing to the processor device that has the least value M and enqueues the queue 4. When either processing is complete, the means 3 records the processing time and dequeues the processing unit from the queue 4. A load dispersion means 1 consists of the means 2 and 3 and a queue 4 and knows the load of other processors with no communication and also can disperse the processing.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平9－288648

(43)公開日　平成９年(1997)11月４日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ	15/16	３７０		Ｇ０６Ｆ 15/16	３７０Ｎ	
	13/00	３５７		13/00	３５７Ｚ	

審査請求　未請求　請求項の数７　ＯＬ　（全 12 頁）

(21)出願番号　特願平8－98174

(22)出願日　平成８年(1996)４月19日

(71)出願人　000005821
松下電器産業株式会社
大阪府門真市大字門真1006番地

(72)発明者　若谷　彰良
大阪府門真市大字門真1006番地　松下電器産業株式会社内

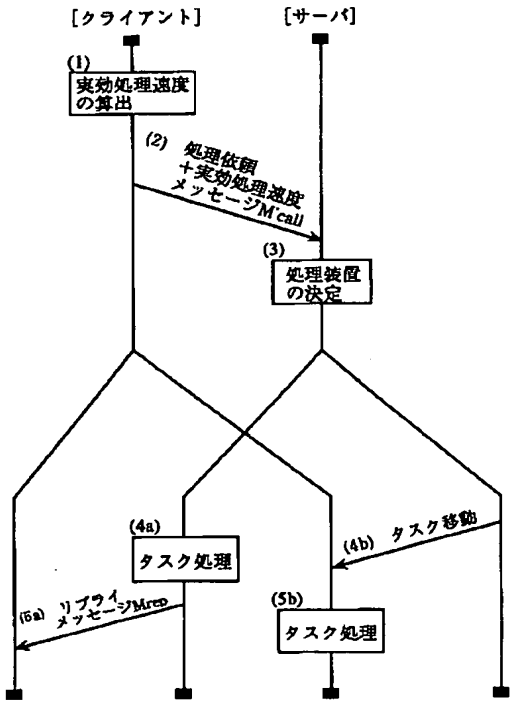
(74)代理人　弁理士　中島　司朗

(54)【発明の名称】　ネットワークにおける分散処理方法及びそのシステム

(57)【要約】

【課題】　ネットワークにおける負荷状況の変動に対応した効率的な分散処理方式を提供する。

【解決手段】　クライアント１は、自らの実効処理速度を算出し（手順１）、その情報と共にサーバ１１に対してサーバ１１が備えるタスクの処理を依頼する（手順２）。サーバ１１は、受信したクライアントと処理速度と自らの処理速度に基づいて、そのタスクに係るターンアラウンド時間を短縮するには、いずれの装置がそのタスクを処理すべきか判断する（手順３）。その判断により、サーバ自らがそのタスクを処理するか（手順４ａ、５ａ）、又は、そのタスクをクライアントに移動した後に（手順４ｂ）クライアントが処理する（手順５ｂ）かが行われる。



【特許請求の範囲】
【請求項１】　複数のタスクを並行処理し得る複数の端末装置が通信路に接続されたネットワークにおいて、一の端末装置（クライアント）が他の一の端末装置（サーバ）に備えられた特定タスクの処理結果を利用したい場合における分散処理方法であって、
前記クライアントが前記結果を利用したい旨の要求を発すると、前記特定タスクに係るターンアラウンド時間を小さくするためにはサーバ又は前記クライアントのいずれがその特定タスクを処理すべきかを判断する第１ステップと、
サーバが処理すべきと判断した場合には、そのサーバは前記特定タスクを処理しその結果を前記クライアントに報告し、一方、前記クライアントが処理すべきと判断した場合には、前記サーバは前記特定タスクを前記クライアントに移動しそのクライアントがその特定タスクを処理する第２ステップとからなることを特徴とする分散処理方法。
【請求項２】　前記第１ステップは、
前記サーバにより行われ、
現時点における前記サーバ及び前記クライアントの実効処理速度を決定する決定ステップと、
決定した実効処理速度に基づいて、前記サーバが前記特定タスクを処理しその結果を前記クライアントに報告した場合に要する第１の時間と、前記サーバから前記クライアントに前記特定タスクを移動しそのクライアントがその特定タスクを処理した場合に要する第２の時間とを求めて比較することにより前記判断をする判断ステップとからなることを特徴とする請求項１記載の分散処理方法。
【請求項３】　前記決定ステップにおいては、前記サーバ及び前記クライアントの実効処理速度Ｒs、Ｒcは、並行処理されているそれぞれのタスク数Ｎs、Ｎcを検出し、それぞれの最大処理速度Ｒsmax、Ｒcmaxを用いて、
式Ｒs＝Ｒsmax／（Ｎs＋１）　及び
Ｒc＝Ｒcmax／（Ｎc＋１）
により決定し、
前記判断ステップにおいては、前記第１の時間Ｔ1及び第２の時間Ｔ2は、前記特定タスクの処理に伴う仕事量をＭtask、前記通信路の通信速度をＲt、前記報告に伴う通信量をＭrep、前記タスク移動に伴う通信量をＭmigとすると、式Ｔ1＝Ｍtask／Ｒs＋Ｍrep／Ｒt　及び
Ｔ2＝Ｍmig／Ｒt＋Ｍtask／Ｒc
により算出することを特徴とする請求項２記載の分散処理方法。
【請求項４】　前記決定ステップにおける前記クライアントの実効処理速度の決定は、そのクライアントが現時点における自らの実効処理速度を算出し、その実効処理速度を示す情報を前記要求と同時に前記サーバに通知し、その通知を受けた前記サーバにより行われることを特徴とする請求項２記載の分散処理方法。
【請求項５】　前記検出ステップにおける前記クライアントの実効処理速度の決定は、前記要求を受けた前記サーバが前記クライアントに問い合わせ、そのクライアントが現時点における自らの実効処理速度を算出して返答し、その返答を受けた前記サーバにより行われることを特徴とする請求項２記載の分散処理方法。
【請求項６】　特定タスクを有するサーバとその特定タスクの処理結果を利用するクライアントとが通信路を介して接続された分散処理システムであって、
前記クライアントは、
前記サーバに前記結果を利用したい旨の要求をする手段と、
現時点における自らの実効処理速度を算出して前記サーバに通知する第１算出手段と、
前記サーバから前記特定タスクが移動されてきた場合には、その特定タスクを処理する手段とを備え、
前記サーバは、
現時点における自らの実効処理速度を算出する第２算出手段と、
前記要求を受けると、前記クライアント及び前記サーバで算出された実効処理速度に基づいて、サーバ自らが前記特定タスクを処理しその結果を前記クライアントに報告した場合に要する第１の時間と、前記特定タスクを前記クライアントに移動しそのクライアントがその特定タスクを処理した場合に要する第２の時間とを求めて比較する比較手段と、
前記第１の時間が小さい場合には、サーバ自らが前記特定タスクを処理しその結果を前記クライアントに報告し、一方、前記第２の時間が小さい場合には、前記特定タスクを前記クライアントに移動する手段とを備えることを特徴とする分散処理システム。
【請求項７】　前記第１算出手段は、
前記クライアントの最大処理速度Ｒcmaxを予め記憶する第１記憶手段と、
現時点において並行処理しているタスク数Ｎcを検出する第１検出手段と、
前記第１記憶手段から読み出した最大処理速度Ｒcmaxと前記第１検出手段により検出されたタスク数Ｎcとから、前記実効処理速度Ｒcを、式
Ｒc＝Ｒcmax／（Ｎc＋１）
により算出する第１速度算出手段とからなり、
前記第２算出手段は、
前記サーバの最大処理速度Ｒsmaxを予め記憶している第２記憶手段と、
現時点において並行処理しているタスク数Ｎsを検出する第２検出手段と、
前記記憶手段から読み出した最大処理速度Ｒsmaxと前記検出手段により検出されたタスク数Ｎcとから、前記実効処理速度Ｒsを、式

Ｒｓ＝Ｒｓｍａｘ／（Ｎｓ＋１）
により算出する第２速度算出手段とからなり、
前記比較手段は、
前記特定タスクの処理に伴う仕事量Ｍｔａｓｋ、前記通信路の通信速度Ｒｔ、前記報告に伴う通信量Ｍｒｅｐ及び前記タスク移動に伴う通信量Ｍｍｉｇを予め記憶している第３記憶手段と、
前記要求を受けると、前記第３記憶手段から読み出した値並びに前記第１速度算出手段及び前記第２速度算出手段により算出された実効処理速度Ｒｃ、Ｒｓを用いて、前記第１の時間Ｔ1及び第２の時間Ｔ2を、式Ｔ1＝Ｍｔａｓｋ／Ｒｓ＋Ｍｒｅｐ／Ｒｔ　及び
Ｔ2＝Ｍｍｉｇ／Ｒｔ＋Ｍｔａｓｋ／Ｒｃ
により算出する時間算出手段と、
前記時間算出手段により算出された第１の時間Ｔ1及び第２の時間Ｔ2を比較する手段とからなることを特徴とする請求項６記載の分散処理システム。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、クライアントとサーバからなる分散処理方法及びそのシステムに関し、特に、タスク処理のターンアラウンド時間を短縮するための分散処理方式に関する。
【０００２】
【従来の技術】近年、インタネット等に代表されるネットワーク通信技術の発達及びワークステーション等に代表される端末装置の高機能化に伴い、クライアント・サーバ形式による分散処理システムが一般化してきた。クライアント・サーバ形式の分散処理システムを実現する第１の従来方式として、遠隔手続き呼出しによる方式（Ｒｅｍｏｔｅ Ｐｒｏｃｅｄｕｒｅ Ｃａｌｌ、以下「ＲＰＣ方式」という。）がある。例えば、"Concurrent Systems", Addison-Wesley,1992に記載されている。ここで、クライアントとは、ある特定のタスク（以下、「特定タスク」という。）の処理を依頼するネットワーク上の端末をいい、サーバとは、その依頼を受けたネットワーク上の端末をいうが、ここでは必ずしもサーバがその特定タスクの処理を実行するとは限らないとする。
【０００３】図９（ａ）はＲＰＣ方式におけるクライアントとサーバとの通信のやりとりを示す説明図であり、図９（ｂ）はそのシーケンス図である。この方式は、大きく３つの手順からなる。即ち、（１）クライアントは、特定タスクの処理を依頼するためにサーバに処理依頼メッセージを送る。（２）サーバは、依頼されたその特定タスクを実行する。（３）実行し終えると、サーバは、その結果を通知するためにクライアントにリプライメッセージを返す。
【０００４】例えば、クライアントがサーバに対して、サーバが持つデータベースに検索処理およびその結果をグラフイメージで返す処理を依頼し、その生成されたイメージ情報をクライアントが受け取る場合が該当する。この方式の長所は、比較的低価格の端末と通信手段によってシステム全体としては高度な処理が実行される点にある。
【０００５】クライアント・サーバ形式の分散処理システムを実現する第２の従来方式として、タスク移動（マイグレーション）による方式がある。例えば、SUN microsystems社が提唱しているJAVA（"Hooked on Java", Addison-Wesley, 1995 に記載）やNetscape社で提案されているもの等（以下、この方式を「ＪＡＶＡ方式」という。）がそうである。
【０００６】図１０（ａ）はＪＡＶＡ方式におけるクライアントとサーバとの通信のやりとりを示す説明図であり、図１０（ｂ）はそのシーケンス図である。この方式も、大きく３つの手順からなる。即ち、（１）クライアントは、特定タスクの処理を依頼するためにサーバに処理依頼メッセージを送る。（２）サーバは、処理手続きと必要データをパッケージした特定タスクの移動を行う。即ち、特定タスク本体をメッセージとして構築し、リプライメッセージとしてクライアントに返信する。（３）その特定タスクを受け取ったクライアントは、自らその特定タスクを実行する。
【０００７】この方式の長所は、クライアントにタスク処理を行わせることにより、処理が集中しがちなサーバでの負荷を軽減させることができ、これによってシステム全体における負荷分散が図れるという点にある。
【０００８】
【発明が解決しようとする課題】しかしながら、上記の従来方式では、クライアントやサーバでの負荷の変動に伴い、必ずしも、ターンアラウンド時間、即ち、処理の依頼が発生してからその処理結果を利用できる状態になるまでの時間が最小になるとは限らないという問題点がある。
【０００９】図１１は、その問題点を説明するためのシステム構成を示すモデル図である。本図に示されるように、いま、システムの処理性能やデータ転送速度、負荷状況をモデル化することにより、以下の通り、各分散処理方式におけるターンアラウンド時間を評価する。まず、各端末装置ごとに、現在処理されているタスクに新たに１個のタスクが加わった場合に、各タスクが処理される速度、即ち、実効処理速度を定義する。
【００１０】クライアントの処理速度の最大値をＲｃｍａｘ（仕事単位／秒）、クライアントで並行処理しているタスク数をＮｃ（個）とすると、クライアントの実効処理速度Ｒｃ（仕事単位／秒・タスク）は、
Ｒｃ＝Ｒｃｍａｘ／（Ｎｃ＋１）　・・・　（式１）
で表されるものとし、同様に、サーバの処理速度の最大値をＲｓｍａｘ（仕事単位／秒）、サーバで並行処理しているタスク数をＮｓ（個）とすると、サーバの実効処理速度Ｒｓ（仕事単位／秒・タスク）は、

Ｒｓ＝Ｒｓｍａｘ／（Ｎｓ＋１）　・・・　（式２）
で表されるものとする。
【００１１】ここで、最大処理速度Ｒｃｍａｘ、Ｒｓｍａｘとは、各端末装置固有の処理能力を示す単位であり、各装置が有するＣＰＵのSPECmark等に相当し、仕事単位とは、タスクの処理サイズを表す単位であり、実効処理速度が１仕事単位／秒である装置が１秒間に処理できる正規化された演算量に相当する。さらに、特定タスクの仕事量をＭｔａｓｋ（仕事単位）、通信路の通信速度をＲｔ（通信単位／秒）、依頼メッセージの通信量をＭｃａｌｌ（通信単位）、リプライメッセージの通信量をＭｒｅｐ（通信単位）、タスク移動に伴う特定タスクの通信量をＭｍｉｇ（通信単位）と定義する。
【００１２】すると、ＲＰＣ方式におけるターンアラウンド時間Ｔｒｐｃ（秒）は、上述の３つの手順に対応する時間の合計になるので、

Ｔｒｐｃ＝Ｍｃａｌｌ／Ｒｔ＋Ｍｔａｓｋ／Ｒｓ＋Ｍｒｅｐ／Ｒｔ　・・・　（式３）

と表され、一方、ＪＡＶＡ方式におけるターンアラウンド時間Ｔｊａｖａ（秒）も、上述の３つの手順に対応する時間の合計になるので、

Ｔｊａｖａ＝Ｍｃａｌｌ／Ｒｔ＋Ｍｍｉｇ／Ｒｔ＋Ｍｔａｓｋ／Ｒｃ　・・・　（式４）

と表される。
【００１３】具体的な数値を用いて両方式でのターンアラウンド時間を評価すると、以下の通りである。今、システムの性能として、クライアントの処理速度の最大値Ｒｃｍａｘが４０、サーバの処理速度の最大値Ｒｓｍａｘが１００、通信路の通信速度Ｒｔが５であり、各通信量・仕事量として、特定タスクの仕事量Ｍｔａｓｋが２００、依頼メッセージの通信量Ｍｃａｌｌが５、リプライメッセージの通信量Ｍｒｅｐが５、タスク移動に伴う特定タスクの通信量Ｍｍｉｇが１００であるとする。
【００１４】そして、第１の負荷状況として、クライアントで並行処理しているタスク数Ｎｃが１、サーバで並行処理しているタスク数Ｎｓが１９の場合を考える。式１及び式２より、クライアント及びサーバの実効処理速度Ｒｃ及びＲｓは、
Ｒｃ＝４０／（１＋１）＝２０
Ｒｓ＝１００／（１９＋１）＝５
となるので、式３及び式４より、ＲＰＣ方式及びＪＡＶＡ方式におけるターンアラウンド時間Ｔｒｐｃ及びＴｊａｖａは、
Ｔｒｐｃ＝５／５＋２００／５＋５／５＝４２（秒）
Ｔｊａｖａ＝５／５＋１００／５＋２００／２０＝３１（秒）
となる。図１２は、この結果をタイムチャートで示したものである。
【００１５】以上より、この第１の負荷状況においては、ＪＡＶＡ方式による場合の方がＲＰＣ方式に比べてターンアラウンド時間が小さくなるので、ＪＡＶＡ方式がより好ましい分散処理方式と言える。続いて、サーバで並行処理しているタスク数Ｎｓがそれまでの１９から９に減少した第２の負荷状況を考える。
【００１６】第１の負荷状況の場合と同様に算出すると、クライアントの実効処理速度Ｒｃは２０のまま変動しないが、サーバの実効処理速度Ｒｓは、
Ｒｓ＝１００／（９＋１）＝１０
と増加するので、ＲＰＣ方式におけるターンアラウンド時間Ｔｒｐｃは、
Ｔｒｐｃ＝５／５＋２００／１０＋５／５＝２２（秒）
と減少する。一方、ＪＡＶＡ方式におけるターンアラウンド時間Ｔｊａｖａは３１秒のまま変動しない。図１３は、この結果をタイムチャートで示したものである。
【００１７】以上より、この第２の負荷状況においては、第１の負荷状況の場合と異なり、ＲＰＣ方式による場合の方がＪＡＶＡ方式に比べてターンアラウンド時間が小さくなるので、ＲＰＣ方式がより好ましい分散処理方式と言える。このように、システムの負荷状況の変動に伴い、両方式におけるターンアラウンド時間の大小関係が反転するが、従来のいずれの方式であっても、負荷の変動には依存しない固定的な方式であるために、必ずしも他方の方式よりも小さいターンアラウンド時間が提供されるとは言えない。即ち、負荷の状況によっては、急激にそのパフォーマンスが落ちるという問題点がある。
【００１８】そこで、本発明は、上記問題点を解決するためになされたものであり、システムの負荷状況が変動する場合であっても最小のターンアラウンド時間が得られる効率的な分散処理方法及びそのシステムを提供することを目的とする。
【００１９】
【課題を解決するための手段】上記目的を達成するために、本発明に係る分散処理方法及びシステムによれば、クライアントがサーバに備えられた特定タスクの処理結果を利用したい旨の要求を発すると、先ず、前記特定タスクに係るターンアラウンド時間を小さくするためにはサーバ又はクライアントのいずれがその特定タスクを処理すべきかが判断され、次に、サーバが処理すべきと判断された場合にはその特定タスクはサーバによって処理された後にその結果が前記クライアントに報告され、一方、前記クライアントが処理すべきと判断された場合にはその特定タスクはサーバからクライアントに移動された後にその特定タスクはクライアントによって処理される。
【００２０】これにより、クライアントからの処理要求が発生する度に、その処理に要するターンアラウンド時間をより小さくするための処理方式が採用されるので、サーバ又はクライアントのいずれかによって固定的に処

理されていた従来方式に比べ、そのターンアラウンド時間が短縮され、システムの負荷状況の変動に対応した効率的な分散処理システムが実現される。

【００２１】

【発明の実施の形態】以下、本発明の実施の形態について図面を用いて詳細に説明する。

（実施の形態１）図１は、実施の形態１に係る分散処理システムのハードウェア構成を示すブロック図であり、ネットワークシステム全体の中の関連する２台の端末装置１、１１のみを示している。

【００２２】本システムは、通信路２０を介して接続されたクライアント１とサーバ１１から構成される。クライアント１及びサーバ１１は、それぞれプロセッサ２、１２、メモリ３、１３及び通信部４、１４から構成される。プロセッサ２、１２は、ＣＰＵ等であり、装置全体を制御したり複数のタスクを並列処理したりする。

【００２３】メモリ３、１３は、ＲＯＭ、ＲＡＭ及びハードディスク等からなり、後述する制御プログラム、システム性能パラメータ、タスク等を予め保持し、また、処理されるタスクの作業域としても用いられる。通信部４、１４は、イーサネット用のインタフェースカード等であり、通信路２０と装置１、１１との中継をする。

【００２４】なお、クライアント１及びサーバ１１は、汎用のワークステーション等の端末装置のハードウェア構成と同じであるが、メモリ３、１３に本実施形態を実現する固有の制御プログラム等を保持している点で汎用の端末装置と異なる。以上のように構成された分散処理システムにおける動作について説明する。ここで、従来技術での説明と同様の状況を考える。即ち、クライアント１は、サーバ１１のメモリ１３に保持されている特定タスクの処理結果を得たいとする。この場合における分散処理方式を説明する。

【００２５】図２は、本実施形態の分散処理方式におけるクライアント１とサーバ１１との通信のやりとりを示すシーケンス図であり、従来の図９（ｂ）及び図１０（ｂ）に対応する。このシーケンス図は、２つの異なる場合における流れを同時に示している。即ち、以下の手順１～３、４ａ、５ａからなる場合と、手順１～３、４ｂ、５ｂからなる２つの場合である。

（手順１）まず、クライアント１は、特定タスクの処理の依頼に先立ち、自らの実効処理速度を算出する。

【００２６】具体的には、プロセッサ２は、自らの処理負荷、即ち、メモリ３にロードされ並列処理されているタスクの数Ncを検出し、さらに、メモリ３に格納されている最大処理速度Rcmaxを読み出し、式１に従って、実効処理速度Rcを算出する。

（手順２）続いて、クライアント１は、算出した実効処理速度Rcとサーバ１１が保持する特定タスクの処理結果を利用したい旨とを一つのメッセージM'callにしてサーバ１１に送信する。

【００２７】具体的には、プロセッサ２は、送信元、即ち、クライアント１のネットワークアドレス、サーバが有する特定タスクを識別する情報等を通信部４を介してサーバ１１に送信する。

（手順３）メッセージM'callを受信したサーバ１１は、その特定タスクに係るターンアラウンド時間を短縮するには、その特定タスクを自ら処理すべきかクライアント１が処理すべきかを判断する。

【００２８】具体的には、プロセッサ１２は、通信部１４を介して受信したメッセージM'callにより特定された特定タスクが保持されたメモリ１３を参照することにより、この特定タスクの仕事量Mtask、タスク移動に伴う通信量Mmig及びリプライメッセージの通信量Mrepを決定すると共に、そのメッセージM'callに含まれるクライアント１の実効処理速度Rcを抽出する。

【００２９】続いて、プロセッサ１２は、通信部１４による定期的な通信路２０の観察に基づいて、現在における通信路２０の平均通信速度Rtを検出する。そして、プロセッサ１２は、クライアント１の場合と同様にして、自らの処理負荷、即ち、実効処理速度Rsを算出する。以上の値に基づいて、プロセッサ１２は、式３及び式４のそれぞれの第２項及び第３項からなる以下の２種類の時間T1、T2を算出し比較する。

【００３０】

$$T1 = Mtask / Rs + Mrep / Rt \quad \cdots \quad （式５）$$

$$T2 = Mmig / Rt + Mtask / Rc \quad \cdots \quad （式６）$$

なお、式５で表されるT1と式６で表されるT2との大小関係は、式３で表されるＲＰＣ方式によるターンアラウンド時間Trpcと式４で表されるＪＡＶＡ方式によるターンアラウンド時間Tjavaとの大小関係と同じになる。なぜなら、式３と式４のそれぞれの第１項は共通だからである。

【００３１】比較の結果、T1＜T2の場合には、サーバ１１は、サーバ自らこの特定タスクを処理すべきと判断し、以下の手順４ａ、５ａを実行する。

（手順４ａ）サーバ１１は、特定タスクを処理する。具体的には、プロセッサ１２は、クライアント１から依頼された特定タスクをメモリ１３の保持部から作業用領域に移し、既に実行しているタスクに加えてタイムシェアリングにより実行する。

（手順５ａ）特定タスクの処理を終えると、サーバ１１は、その処理によって得られた結果をリプライメッセージとしてクライアント１に返信する。

【００３２】具体的には、プロセッサ１２は、クライアント１からの要求に対する返答として、その特定タスクの実行により得られたデータを通信部１４を介してクライアント１に送信する。以上の手順１～３、４ａ、５ａにより、クライアント１から依頼された特定タスクはサーバ１１によって処理されるので、結果的に従来のＲＰＣ方式と同様の分散処理方式が行われたことになる。

【００３３】一方、手順３での比較において、T1≧T2の場合には、サーバ１１は、サーバ１１自らではなくクライアント１が特定タスクを処理すべきと判断し、上記手順４ａ、５ａに代えて以下の手順４ｂ、５ｂを実行する。
（手順４ｂ）サーバ１１は、特定タスクをクライアント１に移動する。
【００３４】具体的には、プロセッサ１２は、クライアント１から依頼された特定タスクをメモリ１３の保持部から読み出し、通信部１４を介してクライアント１に転送する。なお、ここで転送される情報には、特定タスク自体、即ち、処理手続き及び必要な関連データだけでなく、手順２によるクライアント１からの要求に対する返答としてタスク移動が行われた旨のメッセージ等も含まれる。
（手順５ｂ）その特定タスクを受信したクライアント１は、自らその特定タスクを処理する。
【００３５】具体的には、プロセッサ２は、通信部４を介してメモリ３の作業領域にロードした特定タスクを、既に実行しているタスクに加えてタイムシェアリングにより実行する。以上の手順１～３、４ｂ、５ｂにより、クライアント１から依頼された特定タスクはクライアント１自らによって処理されるので、結果的に従来のＪＡＶＡ方式と同様の分散処理方式が行われたことになる。
【００３６】このように、本方式によれば、システムにおける処理負荷の状況、即ち、クライアント１及びサーバ１１において並列処理されているタスク数を考慮することにより、T1＜T2、即ち、Trpc＜Tjavaと判断された場合には、従来のＲＰＣ方式によるタスク処理が行われ、一方、T1≧T2、即ち、Trpc≧Tjavaと判断された場合には、従来のＪＡＶＡ方式によるタスク処理が行われる。
【００３７】これによって、システムの処理負荷の状況に応じ、ＲＰＣ方式又はＪＡＶＡ方式のうち、より小さいターンアラウンド時間が得られる方式が動的に決定されたことになる。図３は、以上のシーケンスにおけるサーバ１１のみの動作を示すフローチャートである。
【００３８】サーバ１１は、クライアント１から送られてきた処理要求メッセージM'callを受信した後（ステップＳ３１）、通信部１４で通信路２０の通信速度Rtを検出する（ステップＳ３２）。そして、受信したメッセージM'callから、実行すべき特定タスクの内容を決定し、クライアント１の実効処理速度Rcを抽出する（ステップＳ３３）。
【００３９】さらに、サーバ１１の実効処理速度Rsを求め（ステップＳ３４）、タスク移動に伴うタスクの通信量Mmig、処理に必要となる演算数Mtaskおよびタスク処理の終了時にリプライに必要となるメッセージの通信量Mrepを求める（ステップＳ３５）。以上の情報を用い、式５で表されるT1と式６で表されるT2とを比較し（ステップＳ３６）、前者が小さい場合は、サーバ１１内でタスク処理を行なった後に（ステップＳ３７）その結果をクライアント１にリプライし（ステップＳ３８）、そうでない場合は、クライアント１にタスク本体を移動してクライアント１内でその特定タスクの処理を行なうようにする（ステップＳ３９）。
【００４０】図４は、図２に示されたシーケンスにおけるクライアント１のみの動作を示すフローチャートである。まず、クライアント１は、実効処理速度Rcを求め（ステップＳ４１）、それを処理依頼メッセージM'callに付加してサーバ１１に送信する（ステップＳ４２）。
【００４１】続いて、サーバ１１からのリプライを受信し（ステップＳ４３）、それが特定タスク本体であれば（ステップＳ４４）、クライアント１内でその特定タスクを実行し（ステップＳ４５）、そうでなければ、そのリプライを特定タスクの処理結果として利用する。以上のシーケンスについて、従来技術での説明と同様の具体的な数値を用いてその流れを説明する。
【００４２】即ち、システムの性能として、Rcmax＝４０、Rsmax＝１００、Rt＝５であり、通信量・仕事量として、Mtask＝２００、M'call＝５、Mrep＝５、Mmig＝１００であり、第１の負荷状況として、Nc＝１、Ns＝１９の場合を考える。すると、
Rc＝４０／（１＋１）＝２０
Rs＝１００／（１９＋１）＝５
より、
T1＝２００／５＋５／５＝４１
T2＝１００／５＋２００／２０＝３０
となるので、上記手順３においては、T1≧T2と判断され、クライアント１が特定タスクを処理すべきと判断されるので、その後、上記手順４ｂ、５ｂ、即ち、従来のＪＡＶＡ方式が採られる。
【００４３】図５は、この第１の負荷状況における本方式のタイムチャートを、従来の固定的なＲＰＣ方式との比較において示す図である。式３及び式４より、本方式により得られるターンアラウンド時間（３１秒）は、従来方式の場合（４２秒）よりも小さい。続いて、Ns＝９に減少した第２の負荷状況を考える。
【００４４】Rs＝１００／（９＋１）＝１０
に増加するので、
T1＝２００／１０＋５／５＝２１
と減少するが、T2はそのまま（３０）であるので、上記手順３においては、T1＜T2と判断され、クライアント１が特定タスクを処理すべきと判断されるので、その後、上記手順４ａ、５ａ、即ち、従来のＲＰＣ方式が採られる。
【００４５】図６は、この第２の負荷状況における本方式のタイムチャートを、従来の固定的なＪＡＶＡ方式との比較において示す図である。式３及び式４より、本方

式により得られるターンアラウンド時間（２２秒）は、従来方式の場合（３１秒）よりも小さい。以上のように、本方式によれば、タスク処理の要求が発生した時点におけるシステムでの処理負荷の状況が考慮されるので、ＲＰＣ方式又はＪＡＶＡ方式のうち、より小さいターンアラウンド時間が得られる分散処理方式が動的に決定される。

（実施の形態２）次に、本発明の実施の形態２に係る分散処理システムを説明する。

【００４６】本実施形態は、クライアントからの処理依頼が発生してからその処理をすべき装置を決定するまでの手順が実施形態１と異なり、他の点においては実施形態１と同じである。従って、ここでは、実施形態１と異なる点のみ説明する。図７は、本実施形態の分散処理方式におけるクライアント１とサーバ１１との通信のやりとりを示すシーケンス図である。実施形態１における図２と比較して判るように、本実施形態では、実施形態１の手順１、２に代えて異なる手順１～４を採用し、それ以降の手順は同一である。

（手順１）まず、クライアント１は、サーバ１１に対して、サーバ１１が保持する特定タスクの処理結果を利用したい旨のメッセージのみを送信する。ここでは、実施形態１と相違し、このメッセージにはクライアント１の実効処理速度Ｒｃの情報は含まれない。

（手順２）そのメッセージを受信したサーバ１１は、自らの処理負荷が軽くなった頃等を見計らって、クライアント１に実効処理速度Ｒｃを問い合わせる。

（手順３）問い合わせを受けたクライアント１は、実施形態１の場合と同様の方法により現時点での自らの実効処理速度Ｒｃを算出する。

（手順４）そして、クライアント１は、その実効処理速度Ｒｃをサーバ１１に送信する。

（手順５、６ａ、７ａ、６ｂ、７ｂ）これらのクライアント１及びサーバ１１での動作は、それぞれ実施形態１の手順３、４ａ、５ａ、４ｂ、５ｂと同一である。

【００４７】なお、図８は、以上のシーケンスにおけるサーバ１１のみの動作を示すフローチャートであり、実施形態１における図３に対応する。以上のように、本実施形態は、実施形態１と比較し、手順５におけるサーバ１１での判断のために必要とされる情報の準備方法（手順１～４）が異なるだけで、手順５における判断内容やその結果による以降の流れは同一であるので、実施形態１と同様に、システムでの処理負荷の状況を考慮した最適な分散処理方式が動的に行われるのは言うまでもない。

【００４８】また、クライアント１は、処理依頼メッセージを発行する毎に実効処理速度Ｒｃを計算するのではなくサーバから問い合わせられた時のみ計算するので、クライアント１の負荷は小さくなる。さらに、サーバ１１は、自分にとって都合のよいタイミングでクライアント１の実効処理速度Ｒｃを獲得できるので、手順６ａ又は７ｂにおいて実際に特定タスクが処理される時刻に近い時刻において手順５での判断を行うことが可能となり、刻々と変化する負荷状況にあっても、より精度の高い処理方式が決定される。

【００４９】以上、本発明に係る分散処理システムについて、実施形態に基づいて説明したが、本発明はこれら実施形態に限られないことは勿論である。即ち、

（１）上記実施形態においては、プロセッサの処理能力が処理対象となる各タスクに均等に分配されることを前提に、各端末装置の実効処理速度は式１及び式２で表されるとしたが、この実効処理速度は、これらの式で表されるものに限定されない。例えば、優先順位を伴う重み付けされたスケジューリングが行われる端末装置であれば、その重み付けを考慮した式となる。

（２）本分散処理方式における処理装置の決定は、サーバ１１によって行われたが、この装置に限定されるものではなく、例えば、クライアント１や第３の端末装置によって行われるとするシステムとすることも考えられる。

【００５０】

【発明の効果】以上の説明から明らかなように、本発明に係る分散処理方法及びシステムによれば、クライアントがサーバに備えられた特定タスクの処理結果を利用したい旨の要求を発すると、先ず、前記特定タスクに係るターンアラウンド時間を小さくするためにはサーバ又はクライアントのいずれがその特定タスクを処理すべきかが判断され（第１ステップ）、次に、サーバが処理すべきと判断された場合にはその特定タスクはサーバによって処理された後にその結果が前記クライアントに報告され、一方、前記クライアントが処理すべきと判断された場合にはその特定タスクはサーバからクライアントに移動された後にその特定タスクはクライアントによって処理される（第２ステップ）。

【００５１】これにより、クライアントからの処理要求が発生する度に、その処理に要するターンアラウンド時間をより小さくするための処理方式が行われるので、サーバ又はクライアントのいずれかによって固定的に処理されていた従来方式に比べ、システムの負荷状況の変動に対応した効率的な分散処理方法及びシステムが実現されるという効果がある。

【００５２】ここで、サーバ又はクライアントのいずれが処理すべきかを決定する方法として、先ず、その時点におけるサーバ及びクライアントの実効処理速度を検出し、次に、その実効処理速度に基づいて、サーバが特定タスクを処理しその結果を前記クライアントに報告した場合に要する第１の時間と、サーバからクライアントに特定タスクを移動しそのクライアントがその特定タスクを処理した場合に要する第２の時間とを求めて比較することにより行うことができる。

【００５３】また、サーバ及びクライアントの実効処理速度Rs、Rcは、並行処理されているそれぞれのタスク数Ns、Ncを検出し、それぞれの最大処理速度Rsmax、Rcmaxを用いて、式Rs＝Rsmax／（Ns＋１）　及び
Rc＝Rcmax／（Nc＋１）
により決定し、第１の時間T1及び第２の時間T2は、特定タスクの処理に伴う仕事量をMtask、通信路の通信速度をRt、報告に伴う通信量をMrep、タスク移動に伴う通信量をMmigとすると、式T1＝Mtask／Rs＋Mrep／Rt　及び
T2＝Mmig／Rt＋Mtask／Rc
により算出することもできる。

【００５４】これにより、刻々と変化する両装置での処理能力を考慮した柔軟で精度の高いな分散処理方式が可能となる。また、前記第１ステップは、クライアントが現時点における自らの実効処理速度を算出し、その実効処理速度を示す情報を前記要求と同時にサーバに通知し、サーバが前記判断を行うとすることもできる。

【００５５】これによって、クライアントからの処理要求と共にクライアントの処理能力がサーバに同時に通知されるので、この分散処理のための通信トラフィックの増加を最小限に抑えることができる。また、前記第１ステップは、前記要求を受けたサーバがクライアントに問い合わせ、クライアントが現時点における自らの実効処理速度を算出して返答し、サーバが前記判断を行うとすることもできる。

【００５６】これによって、サーバは、自らの都合のよいタイミングでクライアントの処理能力を知ることができるので、処理すべき装置をより的確に決定することができるという効果がある。

【図面の簡単な説明】

【図１】本発明の実施の形態１及び２に係る分散処理システムのハードウェア構成を示すブロック図である。

【図２】実施形態１の分散処理方式におけるクライアント１とサーバ１１との通信のやりとりを示すシーケンス図である。

【図３】図２のシーケンスにおけるサーバ１１のみの動作を示すフローチャートである。

【図４】図２のシーケンスにおけるクライアント１のみの動作を示すフローチャートである。

【図５】実施形態１の第１の負荷状況におけるタイムチャートを、従来の固定的なＲＰＣ方式との比較において示す図である。

【図６】実施形態１の第２の負荷状況におけるタイムチャートを、従来の固定的なＪＡＶＡ方式との比較において示す図である。

【図７】実施形態２の分散処理方式におけるクライアント１とサーバ１１との通信のやりとりを示すシーケンス図である。

【図８】図７のシーケンスにおけるサーバ１１のみの動作を示すフローチャートである。

【図９】図９（ａ）は従来のＲＰＣ方式におけるクライアントとサーバとの通信のやりとりを示す説明図であり、図９（ｂ）はそのシーケンス図である。

【図１０】図１０（ａ）は従来のＪＡＶＡ方式におけるクライアントとサーバとの通信のやりとりを示す説明図であり、図１０（ｂ）はそのシーケンス図である。

【図１１】各分散処理方式におけるターンアラウンド時間を評価するためのシステム構成のモデル図である。

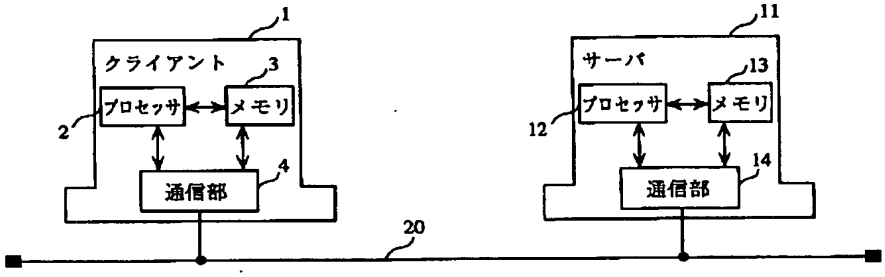
【図１２】ＪＡＶＡ方式による場合の方がＲＰＣ方式に比べてターンアラウンド時間が小さくなる場合の従来例のタイムチャートである。

【図１３】ＲＰＣ方式による場合の方がＪＡＶＡ方式に比べてターンアラウンド時間が小さくなる場合の従来例のタイムチャートである。

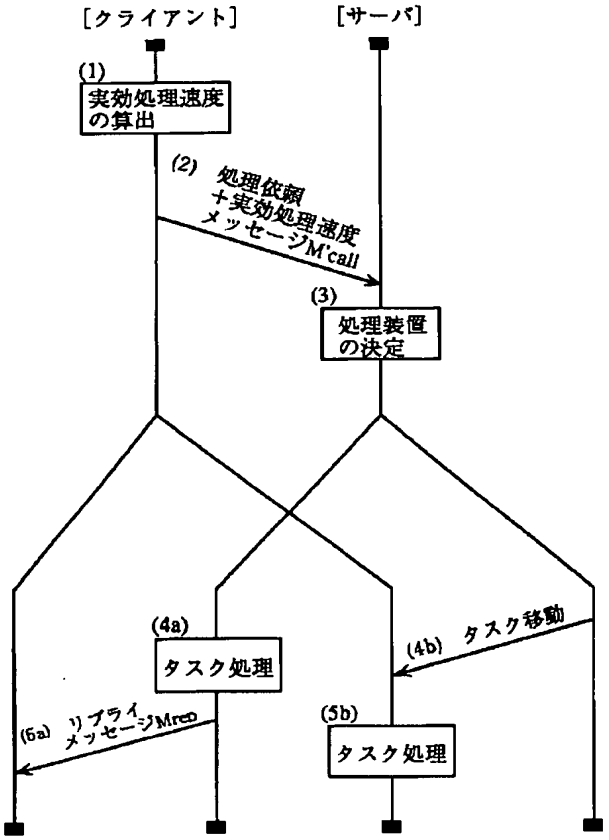
【符号の説明】

１　　クライアント
２　　プロセッサ
３　　メモリ
４　　通信部
１１　　サーバ
１２　　プロセッサ
１３　　メモリ
１４　　通信部
２０　　通信路

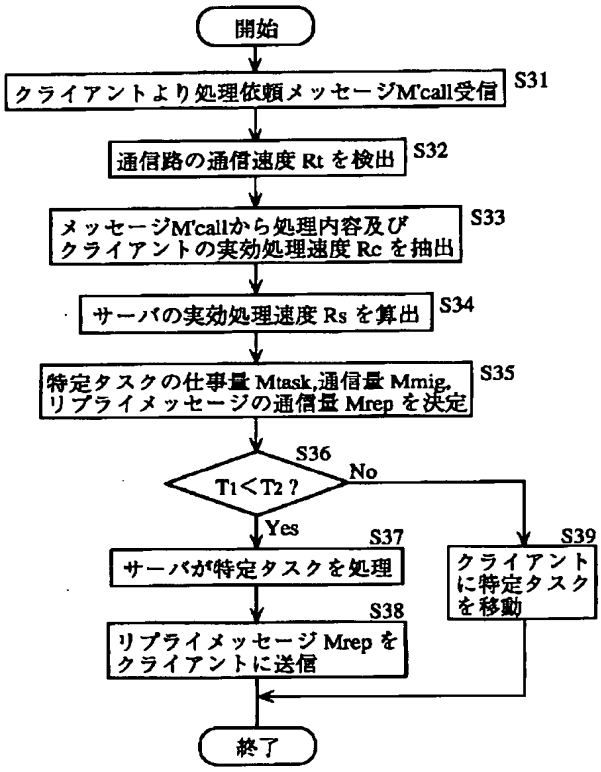
【図１】



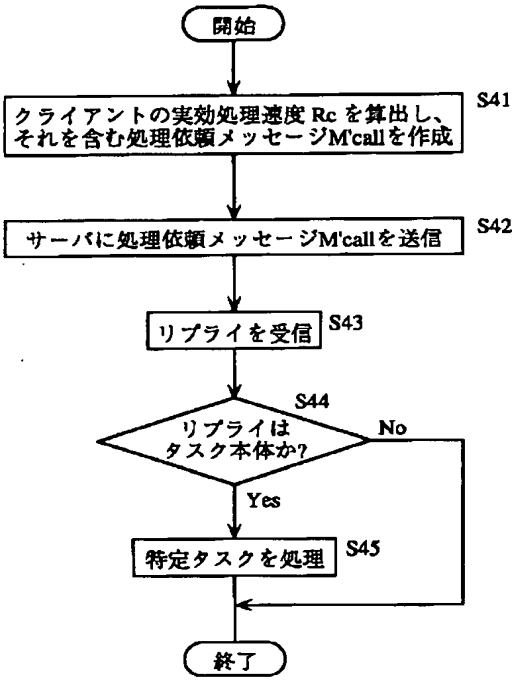
【図2】


[クライアント]
[サーバ]
(1)
実効処理速度の算出
(2)
処理依頼＋実効処理速度メッセージM'call
(3)
処理装置の決定
(4a)
タスク処理
(4b)
タスク移動
(5a)
リプライメッセージMrep
(5b)
タスク処理

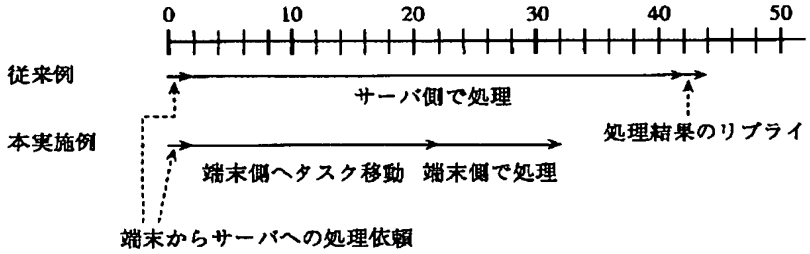
【図3】


開始
クライアントより処理依頼メッセージM'call受信
S31
通信路の通信速度 Rt を検出
S32
メッセージM'callから処理内容及びクライアントの実効処理速度 Rc を抽出
S33
サーバの実効処理速度 Rs を算出
S34
特定タスクの仕事量 Mtask,通信量 Mmig,リプライメッセージの通信量 Mrep を決定
S35
S36
T1＜T2？
No
Yes
S37
サーバが特定タスクを処理
S38
リプライメッセージ Mrep をクライアントに送信
S39
クライアントに特定タスクを移動
終了

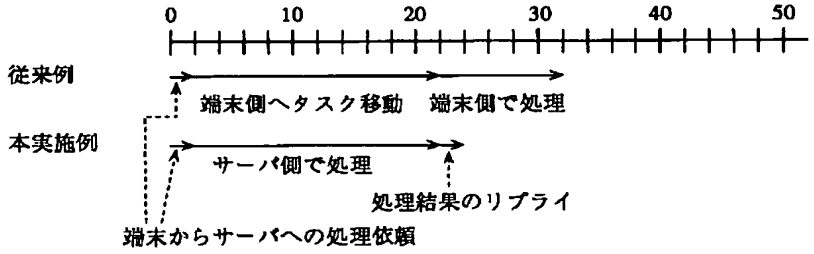
【図4】


開始
クライアントの実効処理速度 Rc を算出し、それを含む処理依頼メッセージM'callを作成
S41
サーバに処理依頼メッセージM'callを送信
S42
リプライを受信
S43
S44
リプライはタスク本体か?
No
Yes
特定タスクを処理
S45
終了

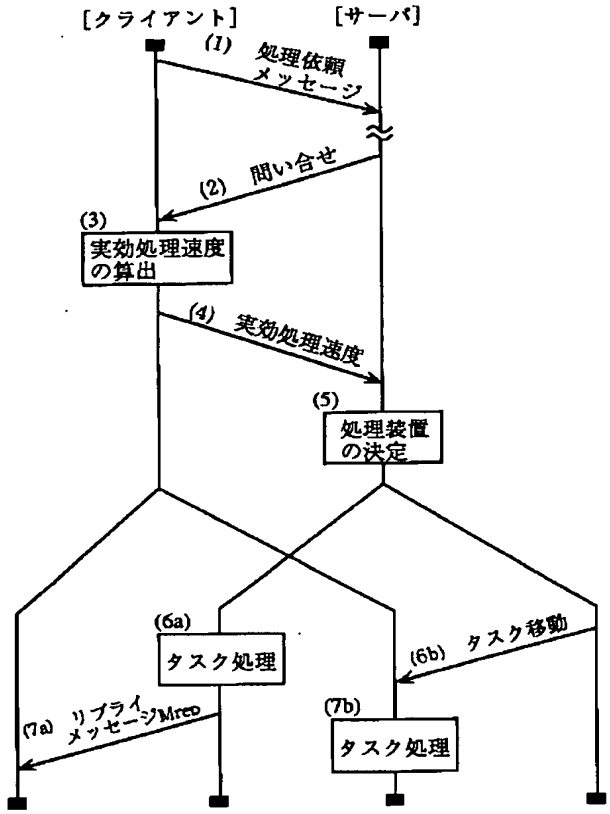
【図5】


0
10
20
30
40
50
従来例
サーバ側で処理
処理結果のリプライ
本実施例
端末側へタスク移動　端末側で処理
端末からサーバへの処理依頼

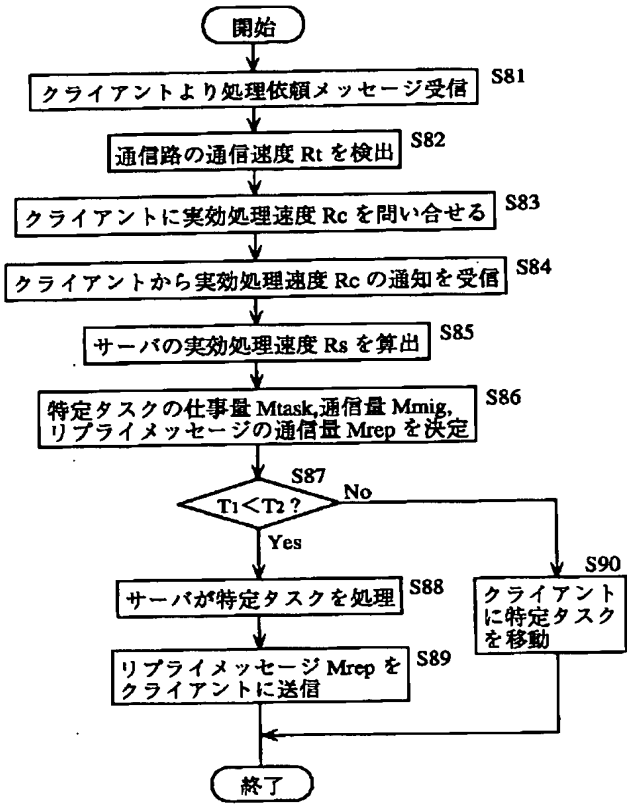
【図6】


0
10
20
30
40
50
従来例
端末側へタスク移動　端末側で処理
本実施例
サーバ側で処理
処理結果のリプライ
端末からサーバへの処理依頼

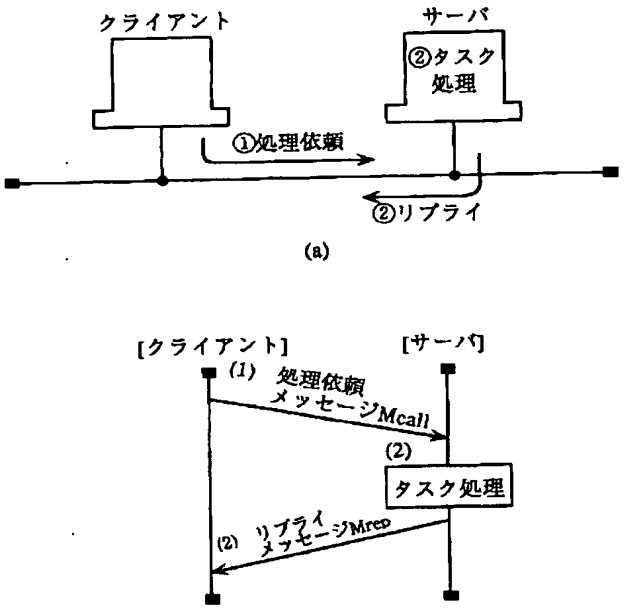
【図7】


[クライアント]
[サーバ]
(1) 処理依頼メッセージ
(2) 問い合せ
(3) 実効処理速度の算出
(4) 実効処理速度
(5) 処理装置の決定
(6a) タスク処理
(6b) タスク移動
(7a) リプライメッセージMrep
(7b) タスク処理

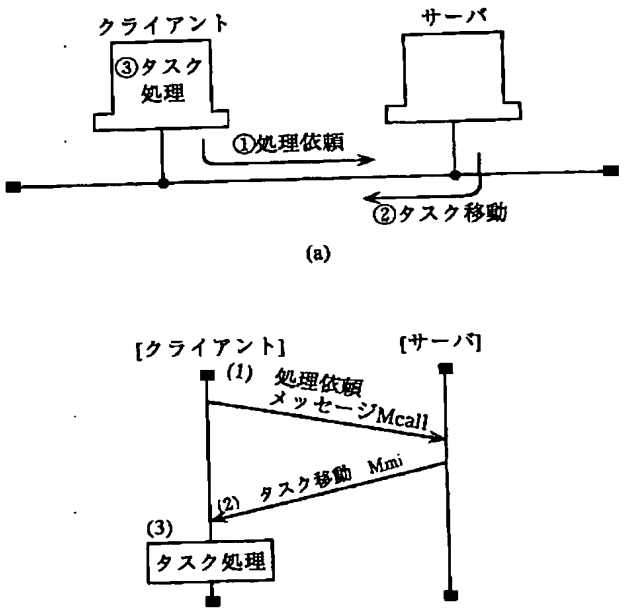
【図8】


開始
クライアントより処理依頼メッセージ受信
S81
通信路の通信速度 Rt を検出
S82
クライアントに実効処理速度 Rc を問い合せる
S83
クライアントから実効処理速度 Rc の通知を受信
S84
サーバの実効処理速度 Rs を算出
S85
特定タスクの仕事量 Mtask,通信量 Mmig, リプライメッセージの通信量 Mrep を決定
S86
S87
T1＜T2？
No
Yes
S90
サーバが特定タスクを処理
S88
クライアントに特定タスクを移動
リプライメッセージ Mrep をクライアントに送信
S89
終了

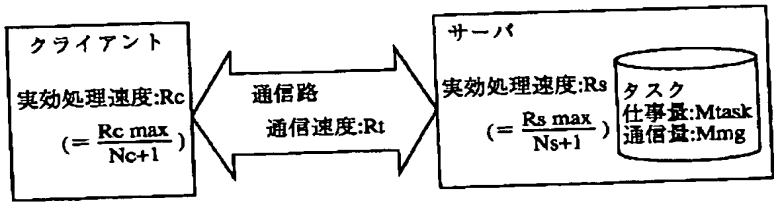
【図９】



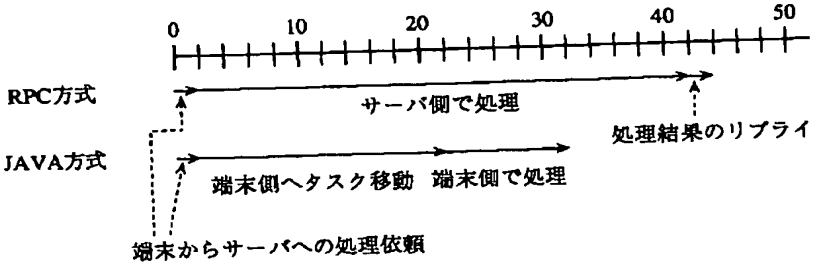
【図１０】



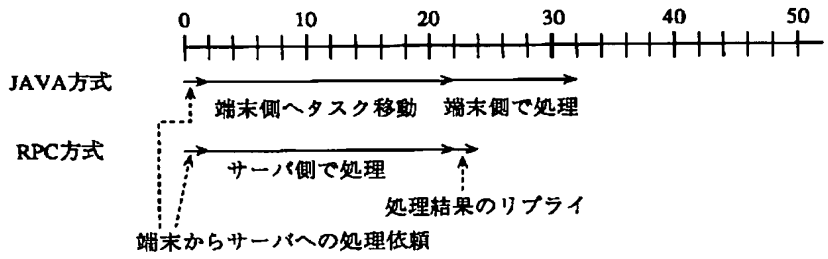
【図１１】



【図１２】



【図１３】


0
10
20
30
40
50
JAVA方式
端末側へタスク移動
端末側で処理
RPC方式
サーバ側で処理
処理結果のリプライ
端末からサーバへの処理依頼